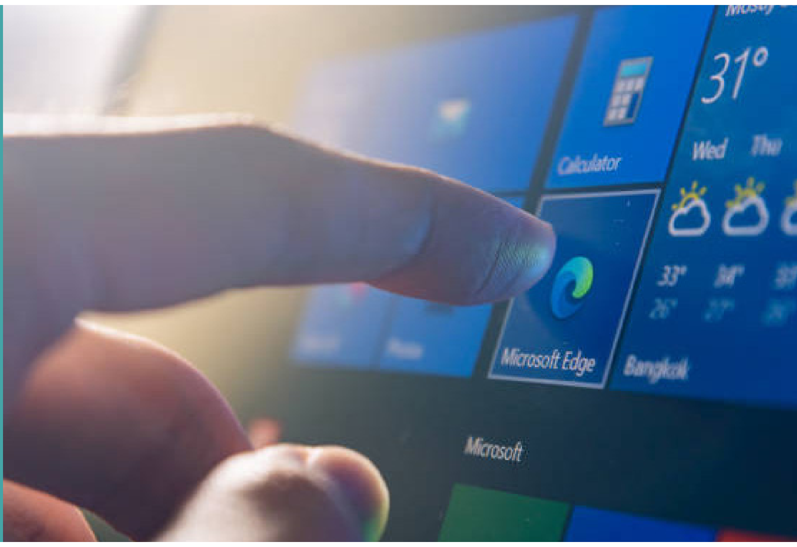
Technical Paper

CCL SOLUTIONS GROUP

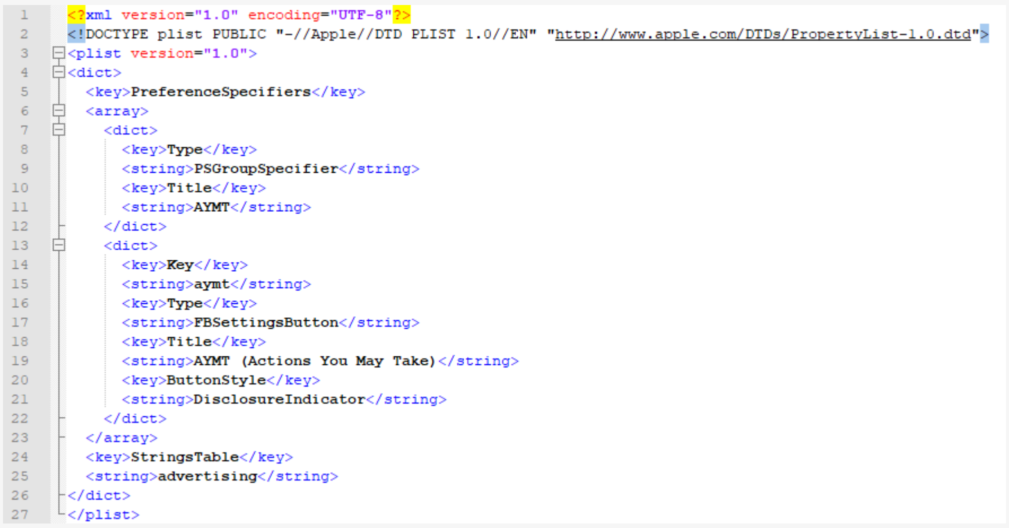
incorporating evidencetalks

Why are there iOS Artefacts in my Windows 10 Applications?

Investigating an unexpected occurence



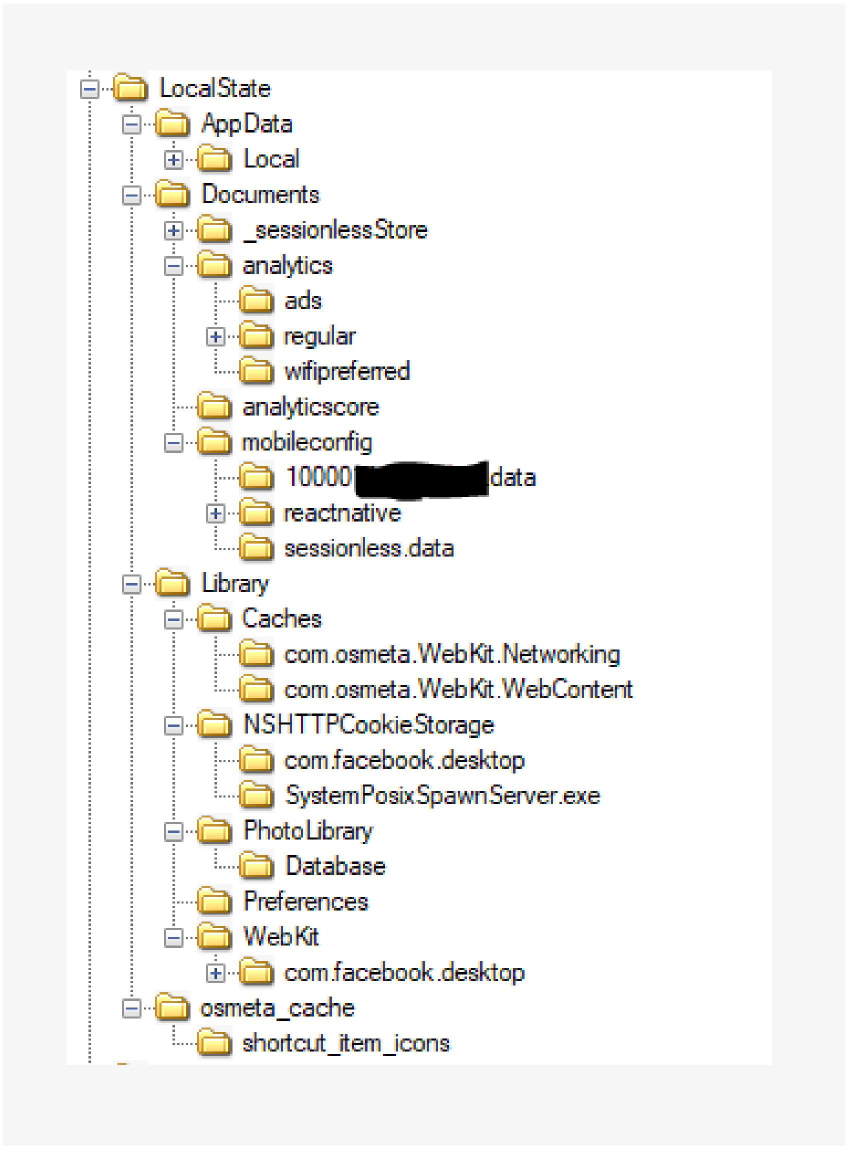
A colleague of mine was diving into the Facebook application on Windows 10 when they noticed something odd: plist files.



```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
  <key>PreferenceSpecifiers</key>
  <array>
    <dict>
      <key>Type</key>
      <string>PSGroupSpecifier</string>
      <key>Title</key>
      <string>AYMT</string>
    </dict>
    <dict>
      <key>Key</key>
      <string>aymt</string>
      <key>Type</key>
      <string>FBSettingsButton</string>
      <key>Title</key>
      <string>AYMT (Actions You May Take)</string>
      <key>ButtonStyle</key>
      <string>DisclosureIndicator</string>
    </dict>
  </array>
  <key>StringsTable</key>
  <string>advertising</string>
</dict>
</plist>
```

For those of you not in the know, plist or "Property List" is a data format used widely across Apple devices, both desktop and embedded devices running iOS and the like. They are widely used to hold configuration information but are often used as a more general data storage format. They come in two flavours: an XML based text format and a more compact binary format – but wait, hold up a second! Let's not get bogged down on what plist files are and focus on the bigger question: what the dickens is this Apple format doing in a Windows application!?

And it doesn't stop with the presence of plists either! Taking a peek at the application data folder `(%APPDATA%\Local\Packages\Facebook.Facebook_8xx8rvfyw5nnt)` and heading into the "LocalState" folder we see this:



```
LocalState
    AppData
        Local
    Documents
        _sessionlessStore
        analytics
            ads
            regular
            wifipreferred
        analyticscore
        mobileconfig
            10000[redacted]data
            reactnative
            sessionless.data
    Library
        Caches
            com.osmeta.WebKit.Networking
            com.osmeta.WebKit.WebContent
        NSHTTPCookieStorage
            com.facebook.desktop
            SystemPosixSpawnServer.exe
        PhotoLibrary
            Database
        Preferences
        WebKit
            com.facebook.desktop
    osmeta_cache
        shortcut_item_icons
```

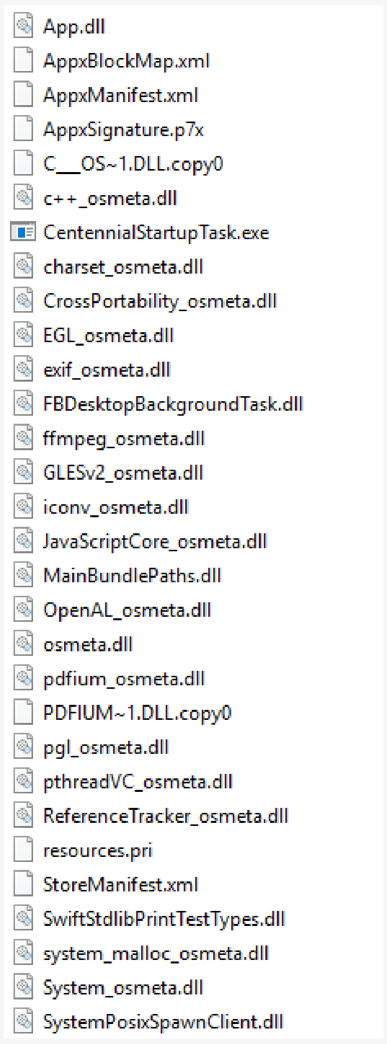
This here looks a heck–of-a–lot like what you'd expect to see in the application folder of an iOS app, especially in the "Library" folder were we even see references to "NSHTTPCookieStorage" which is only a bloomin' Mac API (the files contained therein are packed full of plist goodness too!). What in the world is all of this Mac data doing inside a Windows application?

Turning to the installation folder for the App, which you'll find in "`\Program Files\WindowsApps\Facebook.Facebook_[version goes here]`", we find a range of dlls that the app is making use of, but there is a pattern to a number of the file names – can you spot it?

"`osmeta`" is mentioned in the file names of a number of the dlls. There are even a number of cases where well known libraries (such as ffmpeg and glesv2) have been suffixed by "osmeta" – but what is this mysterious entity?

A little tactical Googling reveals that osmeta is a start-up that Facebook acquired in March 2013. Although osmeta's webpage is long gone, you can still find it on the Internet Archive however it's very light on detail as to what it was that they were building:

"osmeta is a Silicon Valley startup. We are working on really, really interesting software technology. It's big. We will tell you more when the time is right."



App.dll
AppxBlockMap.xml
AppxManifest.xml
AppxSignature.p7x
C__OS~1.DLL.copy0
c++_osmeta.dll
CentennialStartupTask.exe
charset_osmeta.dll
CrossPortability_osmeta.dll
EGL_osmeta.dll
exif_osmeta.dll
FBDesktopBackgroundTask.dll
ffmpeg_osmeta.dll
GLESv2_osmeta.dll
iconv_osmeta.dll
JavaScriptCore_osmeta.dll
MainBundlePaths.dll
OpenAL_osmeta.dll
osmeta.dll
pdfium_osmeta.dll
PDFIUM~1.DLL.copy0
pgl_osmeta.dll
pthreadVC_osmeta.dll
ReferenceTracker_osmeta.dll
resources.pri
StoreManifest.xml
SwiftStdlibPrintTestTypes.dll
system_malloc_osmeta.dll
System_osmeta.dll
SystemPosixSpawnClient.dll

However, there are some clues: multiple mentions of software emulation and virtualisation as well as this image accompanied by the caption: "Incredible possibilities, tremendous opportunities—for developers and users of many, many devices everywhere, including devices in cars, airplanes, and consumer electronics."

Given these clues along with the presence of these "modified" dll files in the installation folder it seems likely that osmeta is providing some kind of interoperability layer, emulation or dare I say: "Meta Operating System" which takes an application designed for one operating system and allows it to run on another. And I'm not the first to reach this **conclusion**.

So as to why we're seeing iOS artefacts all over this Windows 10 app – it's because it is the iOS App.

This is a somewhat unexpected turn of events, but this iOS to Windows translation is not unheard of. In fact, Microsoft themselves have clearly noticed that iOS is a very popular platform for developers of Apps and have started developing Windows Bridge for iOS, also known as "Project Islandwood" and "WinObjC" https://developer.microsoft.com/en-us/windows/bridges/ios, which makes it very easy for developers to quickly port their iOS Apps to Windows 10 – so we might expect to see even more iOS-ness making its way to a Windows 10 machine near you, soon!

Alex Caithness, Principal Analyst (Research & Development)

Aug 2018